A 7/2-Approximation Algorithm for the Maximum Duo-Preservation String Mapping Problem

Nicolas Boria[1], Gianpiero Cabodi[2], Paolo Camurati[3], Marco Palena[4], Paolo Pasini[5], and Stefano Quer[6]

1 **Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, Italy,** nicolas.boria@polito.it
2 **Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, Italy,** gianpiero.cabodi@polito.it
3 **Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, Italy,** paolo.camurati@polito.it
4 **Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, Italy,** marco.palena@polito.it
5 **Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, Italy,** paolo.pasini@polito.it
6 **Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, Italy,** stefano.quer@polito.it

Abstract

This paper presents a simple 7/2-approximation algorithm for the MAX DUO-PRESERVATION STRING MAPPING (MPSM) problem. This problem is complementary to the classical and well studied MIN COMMON STRING PARTITION problem (MCSP), that computes the minimal edit distance between two strings when the only operation allowed is to shift blocks of characters. The algorithm improves on the previously best known 4-approximation algorithm by computing a simple local optimum.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Polynomial approximation, Max Duo-Preservation String Mapping Problem, Min Common String Partition Problem, Local Search

Digital Object Identifier 10.4230/LIPIcs.CPM.2016.11

1 Introduction

Within the field of stringology, string comparison is one of the central problems, as its applications range from data compression to bioinformatics. There are various ways to measure the similarity of two strings, however the most common measure is the so called *edit distance* that counts the minimum number of edit operations that must be performed in order to transform one string into the other. In the specific field of biology, the edit-distance may provide some measure of the kinship between different species based on the similarities of their DNA, as each edit operation can be considered as a single mutation. In data compression, it may help to store efficiently a set of similar yet different data (e.g., different versions of the same object). Indeed, when a set of elements all have a short edit-distance towards a single "base element", an efficient way to compress the whole set of data might be to store only the "base" element of the set, and then record all the other elements as series of edit operations.

Obviously, the concept of edit distance changes definition based on the set of edit operations that are allowed. We tackle the classical case where the only edit operation that

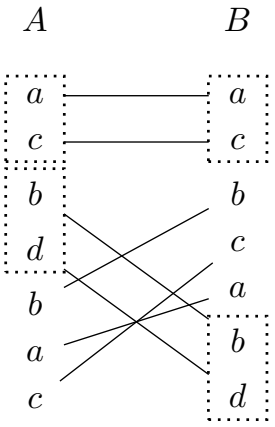

Figure 1 A mapping π that preserves 2 duos.

is allowed is to shift a block of characters, that is, to change the order of the characters in the string by modifying the position of some substring. In this case, the edit distance can be measured by solving the MIN COMMON STRING PARTITION (MCPS).

The MCSP is a fundamental and widely studied problem in the field of string comparison, which applications in the field of bioinformatics are described in [7, 13]. Given a string A let $\mathcal{P}_A$ denote a *partition* of A, that is, a set of substrings whose concatenation results in A. Consider two strings A and B, both with n characters, such that B is a permutation of A. The MCSP Problem introduced in [13] and [18] asks for two partitions $\mathcal{P}_A$ of A and $\mathcal{P}_B$ of B of minimum cardinalities such that $\mathcal{P}_A$ is a permutation of $\mathcal{P}_B$. The $k-$MCSP denotes a natural restriction of the problem where each character of the alphabet has at most k occurrences in each string. In [13], it is shown that this problem is NP-Hard and even APX-Hard. This holds also when the number of occurrences of each character is at most 2, and the result follows from a reduction to MAX INDEPENDENT SET (note that the problem is trivial when the maximal number of occurrences of each character is at most one). Since its introduction in [13], the problem has been intensively studied in various frameworks, such as polynomial approximation [7, 8, 9, 13, 15, 16] and parametric computation [3, 4, 10, 14]. Regarding polynomial approximation, the best results known so far are an $O(\log n \log^* n)$-approximation algorithm for the general version of the problem [9], and an $O(k)$-approximation for $k-$MCSP [16]. Regarding parametric computation, the problem was proved to be Fixed Parameter Tractable (FPT), first with respect to both k and the cardinality ϕ of an optimal partition [3, 10, 14], and more recently, with respect to ϕ only [4].

In [6], the symmetrical (maximization) version of the problem is introduced and denoted by MAX DUO-PRESERVATION STRING MAPPING (MPSM). A *duo* is defined as a couple of consecutive characters in a given string. It is clear that when a couple of partitions $(\mathcal{P}_A, \mathcal{P}_B)$ are a solution for a given instance of MIN COMMON STRING PARTITION that partition A and B into ϕ substrings, this solution is equivalent to a mapping π from characters of A to characters of B that preserves exactly $n - \phi$ duos. A duo is considered *preserved* when its two consecutive characters are mapped to two consecutive characters in the other string. Hence, given two strings A and B, the MPSM problem asks for a mapping π from A to B that preserves a maximum number of duos. An example of mapping that preserves 2 duos is provided in Figure 1.

Reminding that MCSP is NP-Hard [13], its maximization version MPSM is also NP-Hard. However, it is likely that these two problems have different behaviours in terms of ap-

proximation, inapproximability, and parameterized complexity. Among many others, MAX INDEPENDENT SET and MIN VERTEX COVER provide a perfect example of two symmetrical problems having different characteristics: on the one hand, MIN VERTEX COVER is easily 2-approximable in polynomial time by taking all endpoints of a maximal matching [12], and is FPT [5], while on the other hand MAX INDEPENDENT SET is inapproximable within ratio $n^{\varepsilon-1}$ for a given $\varepsilon \in (0,1)$ unless $\mathbf{P} = \mathbf{NP}$ [17], and is $W[1]$-Hard [11].

In [6], some approximation results are presented for MPSM with the following method. A graph problem called CONSTRAINED MAXIMUM INDUCED SUBGRAPH (CMIS) is defined and proved to be a generalization of MPSM. Using a solution to the linear relaxation of CMIS, it is then shown that a randomized rounding provides a k^2 expected approximation ratio for k-CMIS (and thus for k-MPSM), and a 2 expected approximation ratio for 2-CMIS (and thus for 2-MPSM). In [2], these results were improved by introducing and analysing two simple approximation algorithms: the first guarantees a 4-approximation ratio (regardless of the value of k), while the second ensures an approximation ratio $8/5$ when $k = 2$ and ratio 3 when $k = 3$. Moreover, the problem is shown to be APX-Hard. Very recently, the problem was shown to be FPT with respect to the number of duos preserved [1].

In what follows, we present further improvements on the latter results, namely a polynomial $7/2$-approximation algorithm based on a local search technique. In Section 2, we present briefly a graph generalization of MPSM called MAX CONSECUTIVE BIPARTITE MATCHING. Then, we describe our local search algorithm in Section 3 for which we provide complexity analysis (Section 4) and bound on the approximation ratio (Section 5). We finally provide some perspective for future works and possible further improvements on the approximation guarantee in Section 6.

2 Graph translation of the Problem

We are interested in improving on the best known approximation algorithm for MAX DUO-PRESERVATION STRING MAPPING problem, that has approximation ratio 4. In [2], the problem is shown to be a particular case of the following graph problem, which we denote as MAX CONSECUTIVE BIPARTITE MATCHING. Given a bipartite graph where vertices on both sides are ordered : $A = (a_1, ..., a_n)$, $B = (b_1, ..., b_n)$, the MAX CONSECUTIVE BIPARTITE MATCHING problem asks for the maximum matching M such that if $(a_i, b_j) \in M$, then a_{i+1} can only be matched to b_{j+1}, and b_{j+1} can only be matched to a_{i+1}. In other words, sets of matched consecutive vertices on one side must be matched to consecutive vertices on the other side.

Let us recall briefly why MAX DUO-PRESERVATION STRING MAPPING is a particular case of MAX CONSECUTIVE BIPARTITE MATCHING. Strings A and B of any instance of MAX DUO-PRESERVATION STRING MAPPING can be translated as ordered duo sets D^A and D^B (for example, if $A =$ "abc" and $B =$ "bac", then $D^A = ((ab), (bc))$, and $D^B = ((ba), (ac))$).

Consider the bipartite graph $G(I)$ built in the following way (an example is provided in Figure 2):

- each vertex on the left-hand side represents a duo of the set D^A, and each vertex on the right-hand side represents a duo of D^B.
- edges exist between two vertices if and only if they represent the same duo (same couple of characters in the same order)

It is shown in [2] that any feasible solution M for MAX CONSECUTIVE BIPARTITE MATCHING in the graph $G(I)$ yields a mapping π between strings A and B that preserves at least $|M|$ duos (and exactly $|M|$ duos if M is inclusion-wise maximal).

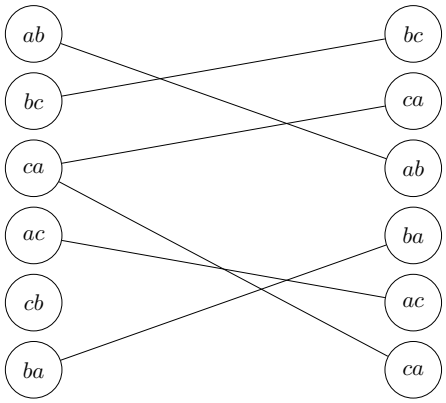

Figure 2 Graph $G(I)$ when I consists of $A =$ "$abcacba$" and $B =$ "$bcabaca$".

Indeed, such a matching can be seen as a partial mapping, and the number of edges in the matching is equal to the number of duos that the mapping preserves. The partial mapping can then be completed in an arbitrary way, since the set of non-mapped characters in A is a permutation of non-mapped vertices in B.

In the rest of the paper, we will refer only to the MAX CONSECUTIVE BIPARTITE MATCHING problem, bearing in mind that any approximation result that holds for MAX CONSECUTIVE BIPARTITE MATCHING also holds MAX DUO-PRESERVATION STRING MAPPING.

We call two edges *conflicting* if they cannot be both part of the same solution, either because they share a common endpoint or because their endpoints are consecutive on one side of the graph but not on the other.

In the following, we present an algorithm that produces such a partial mapping based on local search technique.

3 Local search algorithm

Local search algorithms produce solutions that are defined as local optima. A local optimum of an optimization problem is a solution that is optimal (either maximal or minimal) within a neighbouring set of candidate solutions. Starting from any feasible solution, the algorithm searches an improving solution in the neighbouring set, and repeatedly moves to an improving neighbouring solution as long as such a solution exists. When no improving neighbouring solution can be found, then the current solution is by definition a local optimum.

The quality of the local optimum obviously depends on the definition of the neighbouring set.

We devise a local search algorithm denoted `LOCAL`, which is based on a neighbourhood structure $\mathcal{N}$. Given a matching M that is a feasible solution for the problem, the neighbourhood of M, called $\mathcal{N}(M)$, contains all feasible solutions M' such that $|M \setminus M'| \leq 1$. In other words M' must contain all edges of M apart from possibly one.

While searching for an improving solution in the neighbouring set, the algorithm `LOCAL` will first try to improve the solution without removing any edge from the current solution M. On the one hand, if M is not inclusion-wise maximal, then there is an edge that can be added to the current solution M without having to remove any edge from it. If on the other hand M is inclusion-wise maximal, then the algorithm scans every matching $M \setminus \{v\}$ (for each $v \in M$) and checks if at least two edges can be added to one of these matchings. The pseudocode of algorithm `LOCAL` is provided in Section 4.

Algorithm 1 ALGORITHM LOCAL

Input: $G = (V, E)$
Output: M

```
1:  M = ∅, M' = ∅
2:  if ∃v ∈ E then
3:     M' ← {v}
4:  end if
5:  while M ≠ M' do
6:     M' ← M
7:     for each v ∈ E do
8:        if M ∪ {v} is feasible then
9:           M ← M ∪ {v}
10:          continue
11:       end if
12:    end for
13:    for each v ∈ M do
14:       for each (u, w) ∈ E × E do
15:          if M \ {v} ∪ {u, w} is feasible then
16:             M ← M \ {v} ∪ {u, w}
17:             break
18:          end if
19:       end for
20:    end for
21: end while
22: return M
```

4 Complexity analysis

We prove that the algorithm runs indeed in polynomial time. First of all, even starting from an empty solution, the algorithm will increment the value of its solution by at least one at each step, so that it will conclude after at most $|\text{SOL}| \leq n$ steps.

At each step, the algorithm first scans all edges that are not in SOL and checks if one of them does not conflict with any edge of SOL. This is done in $O(n^2)$ time. If such an edge is found, the current step is finished. Otherwise, for each edge u of SOL, the algorithm considers all sets of at most 6 non-solution edges conflicting with u, and checks if they can be added to the matching $\text{SOL} \setminus \{u\}$ without generating any conflict. This is done in $O(n^6)$ time for each edge u of the current solution: each edge of the solution conflicts with $O(n)$ non-solution edges, so that there are $O(n^6)$ candidate combinations of at most 6 non-solution edges to consider. Considering that, at each step, the current solution has $O(n)$ edges, the complexity of a single step is $O(n^7)$.

In all, the algorithm finishes after at most n steps, each step running in $O(n^7)$ time, so that the overall complexity is $O(n^8)$.

The complexity of LOCAL can actually been brought down to $O(n^4)$ thanks to the following observation. If an improvement incrementing the cardinality of the solution by at least one can be made at some step (by removing an edge u of SOL and adding a set X of at least two non conflicting edges to SOL), then an improvement incrementing this value by exactly one is also possible (by removing the same edge u of SOL and adding exactly any couple of edges of the set X). Thus, instead of scanning all sets of at most 6 non-solution edges conflicting

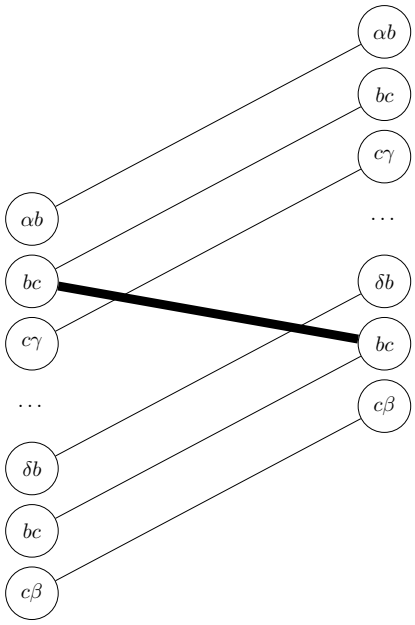

Figure 3 Conflicts among SOL and OPT edges.

with each edge u, it suffices that the algorithm scans only every couple of non-solution edges conflicting with u. If no improving couple can be found, then no improvement of any kind can be made, and the current solution is a local optimum.

5 Approximation analysis

We now prove that, indeed, `LOCAL` improves on the best known 4-approximate algorithm for MAX CONSECUTIVE BIPARTITE MATCHING:

▶ **Theorem 1.** *The algorithm `LOCAL` yields a 3.5 approximation ratio for* MAX CONSECUTIVE BIPARTITE MATCHING *problem.*

Consider that the algorithm `LOCAL` runs on an instance I of MAX CONSECUTIVE BIPARTITE MATCHING and outputs a solution SOL.

The proof is based on counting the conflicts between edges of SOL and edges of an unknown optimal matching OPT. We denote such number of conflicts by C.

On the one hand, a single edge of SOL cannot be conflicting with more than 6 edges of OPT (the worst case is shown in Figure 3). Indeed, on the one hand, any edge u can be in conflict only with edges that share an endpoint with u, or that have an endpoint that is consecutive to an endpoint of u (immediately after or immediately before), which results in no more than 6 possible endpoints for edges conflicting with u (the two endpoints of u, and the four consecutive vertices). On the other hand, any feasible solution including the optimal one can pick at most one edge per vertex of the graph. This gives us the following upper bound on the value of C:

$$C \leq 6|\text{SOL}|\,. \tag{1}$$

We recall that, by definition, there is no solution SOL$'$ in the neighbourhood $\mathcal{N}(\text{SOL})$ of SOL that has more edges than SOL. Hence, given any edge v of SOL the following fact holds:

▶ **Fact 2.** *Let v be an edge of solution* SOL *generated by* `LOCAL`*, and* OPT *be an optimal solution for the problem. There is at most one edge u of* OPT *that conflicts only with v in* SOL*.*

The fact is rather straightforward: suppose that there exist two edges u and t in a solution OPT that both conflict with a single edge v in SOL. The solution $\mathrm{SOL} \setminus \{v\} \cup \{u, t\}$ is an admissible matching in the neighbourhood of SOL and it contains more edges. Hence, `LOCAL` should have picked it instead of SOL.

Let us denote by k_1 the number of edges in OPT that conflict with one edge of SOL only. Fact 2 yields naturally the following bound:

$$k_1 \leq |\mathrm{SOL}|\,. \tag{2}$$

In OPT the remaining $|\mathrm{OPT}| - k_1$ edges conflict with at least 2 edges of SOL, which gives us the following lower bound on the number of conflicts C:

$$C \geq k_1 + 2(|\mathrm{OPT}| - k_1) \geq 2|\mathrm{OPT}| - k_1 \underset{(2)}{\geq} 2|\mathrm{OPT}| - |\mathrm{SOL}|\,. \tag{3}$$

Combining equations (1) and (3), we can easily get the following bound on the approximation ratio of `LOCAL`, which concludes the proof:

$$\frac{\mathrm{OPT}}{\mathrm{SOL}} \leq \frac{7}{2}\,.$$

6 Conclusion and perspectives

We showed that a simple local optimization technique provides a better approximation guarantee than the previously best known algorithm for MPSM. The analysis of more complex local optimums that rely on broader (yet polynomial) definitions of neighbourhood did not lead to immediate further improvements of the approximation guarantee. However, there are strong hints that, in such optimums, the number of edges that conflict with 6 edges of a global optimum is somehow linked to the number of edges of the global optimum conflicting with few edges of the local optimum. Namely, if many edges of the local optimum conflict with 6 edges of the global optimum, then few edges of the global optimum are expected to conflict few edges of the global optimum, resulting in a tighter version of equation 3, bounding for example the value $C(t)$ where t is the number of edges of the local optimum that conflict with 6 edges of the global optimum. Analysing such a bound might eventually lead to further improvements on the approximation ratio.

References

1 S. Beretta, M. Castelli, and R. Dondi. Parameterized Tractability of the Maximum-Duo Preservation String Mapping Problem. *ArXiv e-prints*, December 2015. `arXiv:1512.03220`.

2 N. Boria, A. Kurpisz, S. Leppänen, and M. Mastrolilli. Improved approximation for the maximum duo-preservation string mapping problem. In Dan Brown and Burkhard Morgenstern, editors, *Algorithms in Bioinformatics*, volume 8701 of *Lecture Notes in Computer Science*, pages 14–25. Springer Berlin Heidelberg, 2014. `doi:10.1007/978-3-662-44753-6_2`.

3 L. Bulteau, G. Fertin, C. Komusiewicz, and I. Rusu. A Fixed-Parameter Algorithm for Minimum Common String Partition with Few Duplications. In *WABI*, pages 244–258, 2013. `doi:10.1007/978-3-642-40453-5_19`.

4 L. Bulteau and C. Komusiewicz. Minimum common string partition parameterized by partition size is fixed-parameter tractable. In *SODA*, pages 102–121, 2014. doi:10.1137/1.9781611973402.8.

5 J. Chen, I.A. Kanj, and W. Jia. Vertex Cover: Further Observations and Further Improvements. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, *WG*, volume 1665 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 1999. doi:10.1007/3-540-46784-X_30.

6 W. Chen, Z. Chen, N. F. Samatova, L. Peng, J. Wang, and M. Tang. Solving the maximum duo-preservation string mapping problem with linear programming. *Theoretical Computer Science*, 530(0):1–11, 2014. doi:10.1016/j.tcs.2014.02.017.

7 X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of Orthologous Genes via Genome Rearrangement. *Transactions on Computational Biology and Bioinformatics*, 2(4):302–315, 2005. doi:10.1145/1100863.1100950.

8 M. Chrobak, P. Kolman, and J. Sgall. The Greedy Algorithm for the Minimum Common String Partition Problem. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *APPROX-RANDOM*, volume 3122 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2004. doi:10.1007/978-3-540-27821-4_8.

9 G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Transactions on Algorithms*, 3(1), 2007. doi:10.1145/1219944.1219947.

10 P. Damaschke. Minimum Common String Partition Parameterized. In Keith A. Crandall and Jens Lagergren, editors, *WABI*, volume 5251 of *Lecture Notes in Computer Science*, pages 87–98. Springer, 2008. doi:10.1007/978-3-540-87361-7_8.

11 R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999. 530 pp.

12 M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, CA, 1979.

13 A. Goldstein, P. Kolman, and J. Zheng. Minimum Common String Partition Problem: Hardness and Approximations. In Rudolf Fleischer and Gerhard Trippen, editors, *ISAAC*, volume 3341 of *Lecture Notes in Computer Science*, pages 484–495. Springer, 2004.

14 H. Jiang, B. Zhu, D. Zhu, and H. Zhu. Minimum common string partition revisited. *Journal of Combinatorial Optimization*, 23(4):519–527, 2012. doi:10.1007/s10878-010-9370-2.

15 P. Kolman and T. Walen. Approximating reversal distance for strings with bounded number of duplicates. *Discrete Applied Mathematics*, 155(3):327–336, 2007. doi:10.1016/j.dam.2006.05.011.

16 P. Kolman and T. Walen. Reversal Distance for Strings with Duplicates: Linear Time Approximation using Hitting Set. *Electronic Journal of Combinatorics*, 14(1), 2007. URL: http://www.combinatorics.org/Volume_14/Abstracts/v14i1r50.html.

17 C. Lund and M. Yannakakis. The Approximation of Maximum Subgraph Problems. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *ICALP*, volume 700 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 1993. doi:10.1007/3-540-56939-1_60.

18 K.M. Swenson, M. Marron, J.V. Earnest-DeYoung, and B.M.E. Moret. Approximating the true evolutionary distance between two genomes. *ACM Journal of Experimental Algorithmics*, 12, 2008. doi:10.1145/1227161.1402297.